WEST

Help | Logout | Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences | Cases

Search Results -

Terms	Documents
L20 and extend	7

Database: US Patents Full-Text Database; US Pre-Grant Publication Full-Text Database; JPO Abstracts Database; EPO Abstracts Database; Derwent World Patents Index; IBM Technical Disclosure Bulletins

Search:

Refine Search

Recall Text | Clear

Search History

DATE: Thursday, July 31, 2003 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L21	L20 and extend	7	L21
L20	L19 and macro near language	74	L20
L19	(((717/$)!.CCLS.))	4982	L19
L18	(((717/140)!.CCLS.))	207	L18
L17	(((717/124)!.CCLS.))	247	L17
L16	(((717/122)!.CCLS.))	79	L16
L15	(((717/118)!.CCLS.))	107	L15
L14	(((717/117)!.CCLS.))	48	L14
L13	(((717/116)!.CCLS.))	201	L13
L12	(((717/115)!.CCLS.))	60	L12
L11	(((717/114)!.CCLS.))	164	L11
L10	(((717/106)!.CCLS.))	154	L10
L9	(((717/209)!.CCLS.))	0	L9
L8	(((707/100)!.CCLS.))	1695	L8
L7	(((707/2)!.CCLS.))	1424	L7
L6	(((715/526)!.CCLS.))	278	L6
L5	(((715/513)!.CCLS.))	1030	L5
L4	(((715/500)!.CCLS.))	544	L4
L3	(((715/$)!.CCLS.))	6140	L3
L2	(((707/$)!.CCLS.))	16017	L2
L1	((707/1)!.CCLS.)	2535	L1

END OF SEARCH HISTORY

WEST

Generate Collection Print

L21: Entry 4 of 7 File: USPT Oct 7, 1997

US-PAT-NO: 5675805
DOCUMENT-IDENTIFIER: US 5675805 A

TITLE: Programming system for generating client and server programs from an undistributed application program

DATE-ISSUED: October 7, 1997

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Boldo; Irit	Neve Sheanan			IL
Shani; Uri	Givat Adi			IL
Gold; Israel	Haifa			IL

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 08/ 667314 [PALM]
DATE FILED: June 20, 1996

PARENT-CASE:
This is a continuation of application Ser. No. 08/248,909 filed May 25, 1994 now abandoned.

FOREIGN-APPL-PRIORITY-DATA:

COUNTRY	APPL-NO	APPL-DATE
GB	9316948	August 14, 1993

INT-CL: [06] G06 F 9/45

US-CL-ISSUED: 395/706; 395/684, 395/707, 395/200.03
US-CL-CURRENT: 717/114; 709/330

FIELD-OF-SEARCH: 395/200, 395/706, 395/684, 395/707, 395/200.03

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4916610	April 1990	Bapat	364/300
☐	5075847	December 1991	Fromme	395/700
☐	5185867	February 1993	Ito	395/375
☐	5307490	April 1994	Davidson et al.	395/650
☐	5325524	June 1994	Black et al.	395/600
☐	5339430	August 1994	Lundin et al.	395/700

OTHER PUBLICATIONS

Hofmeister et al, Surgeon: A Packager for Dynamically Reconfigurable Distributed Applications, Configurable Distributed Sys. 1992.
Callahan et al, A Packaging System for Heterogeneous Execution Environments, IEEE Trans. on Software Engr. Jun. 1991, V17, I 6.
Dineen et al, The Network Computing Architecture and System; An Environment for Developing Distributed Applications, Compcon'88 IEEE Conf. pp. 296-299.
Stoyenko, Alexander D., A General Model and Mechanisms for Model-Level Heterogeneous RPC Interoperability, Parallel & Distributed Processing, 1990.
Stoyenko, Alexander D., Supra-RPC: Subprogram Parameters in Remote Procedure Calls, Parallel & Distributed Processing, 1991.
Kishimoto et al, Softon: A Flexible Software Construction Model by Interface Mediation, Compsac 1991.
Shani et al, Distributed-Application Development Tools for DCE/OSF, Services in Distributed & Networked Environments, 1994 conf.
Rancov et al, Context Driven Call: Principles, Globecom '92 IEEE Global Telecommunications Conference, 1992.

ART-UNIT: 236

PRIMARY-EXAMINER: Toplu; Lucien U.

ATTY-AGENT-FIRM: Carwell; Robert M.

ABSTRACT:

A programming aid for generating interface definition files for client server programs. The system and method are responsive to an input interface definition file and one or more input source code files to extract, from one such input source code file, the semantics of procedures therein intended to be called by a remote procedure call, to detect any conflicts between the extracted semantics and the input interface definition file, to generate an output interface definition file and to report said conflicts to a user. The user can thereby develop a correct interface definition file through an iterative process by modifying the output interface definition file and using it again as the input interface definition file.

1 Claims, 7 Drawing figures

WEST

Generate Collection Print

L21: Entry 5 of 7 File: USPT Jun 17, 1997

US-PAT-NO: 5640576
DOCUMENT-IDENTIFIER: US 5640576 A

TITLE: System for generating a program using the language of individuals

DATE-ISSUED: June 17, 1997

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Kobayashi; Kaname	Kawasaki			JP
Kimura; Takahisa	Kawasaki			JP

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Fujitsu Limited	Kawasaki			JP	03

APPL-NO: 08/ 244561 [PALM]
DATE FILED: July 25, 1994

FOREIGN-APPL-PRIORITY-DATA:

COUNTRY	APPL-NO	APPL-DATE
JP	4-263954	October 2, 1992

PCT-DATA:

APPL-NO	DATE-FILED	PUB-NO	PUB-DATE	371-DATE	102(E)-DATE
PCT/JP93/01331	September 17, 1993	WO94/08290	Apr 14, 1994	Jul 25, 1994	Jul 25, 1994

INT-CL: [06] G06 F 17/28

US-CL-ISSUED: 395/759; 395/701, 395/702
US-CL-CURRENT: 704/9; 717/146

FIELD-OF-SEARCH: 364/419.08, 364/419.1, 364/419.17, 395/375, 395/700, 395/600

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5038316	August 1991	Hempleman et al.	364/943.5
☐	5452206	September 1995	Turrietta et al.	364/419.17

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
63-170735	July 1988	JP	
2-294833	December 1990	JP	
3-2923	January 1991	JP	
4-273329	September 1992	JP	

ART-UNIT: 241

PRIMARY-EXAMINER: Hayes; Gail O.

ASSISTANT-EXAMINER: Poinvil; Frantzy

ATTY-AGENT-FIRM: Staas & Halsey

ABSTRACT:

A program generating system uses the language of an individual to write a program in expressions with words of the programmer himself which can easily be understood from the side of an application object, and to generate expressions according to a designated programming language from the program written with the words of the programmer himself. The program generating system has a first processor for transforming a user's linguistic expression into an initial role tree, a second processor for detailing and transforming a vocabulary expression of said initial role tree into a beginning role tree, a third processor for detailing and transforming said beginning role tree into an ending role tree, and a fourth processor for generating a source code of a programming language from said ending role tree. The first processor transforms the user's linguistic expression into a description in the form of the role tree. The second processor applies vocabulary transformation rules to the initial role tree to detail the vocabulary expression. The third processor applies role tree transformation rules to the beginning role tree to detail the beginning role tree. The fourth processor applies a set of patterns and a set of vocabularies to the ending role tree to generate a source code of a programming language.

17 Claims, 81 Drawing figures

WEST

Help | Logout | Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences | Cases

Search Results -

Terms	Documents
runtime near macro	1

Database: US Patents Full-Text Database; US Pre-Grant Publication Full-Text Database; JPO Abstracts Database; EPO Abstracts Database; Derwent World Patents Index; IBM Technical Disclosure Bulletins

Search: Refine Search

Recall Text | Clear

Search History

DATE: Thursday, July 31, 2003 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L4	runtime near macro	1	L4
L3	L1 and compiler near macro	9	L3
L2	L1 and runtime near macro	0	L2
L1	preprocessor near macro	75	L1

END OF SEARCH HISTORY

WEST

Generate Collection | Print

L3: Entry 7 of 9 File: PGPB Feb 13, 2003

DOCUMENT-IDENTIFIER: US 20030033588 A1
TITLE: System, method and article of manufacture for using a library map to create and maintain IP cores effectively

Brief Description of Drawings Paragraph (101):
[0108] FIG. 87E illustrates a method for generating libraries utilizing pre-compiler macros, in accordance with one embodiment of the present invention;

Detail Description Paragraph (2493):
[2607] The system described in this example consists of a ROM containing the program to execute, a RAM containing some scratch variables and a processor that understands 10 opcodes. Each instruction is made up of a 4 bit opcode and a 4 bit operand. The _asm preprocessor macro is the assembler for this language and is used to fill in the entries in the program ROM declaration.

Detail Description Paragraph (4278):
[4391] A serious limitation with preprocessor macros (those defined with #define) is their inability to generate recursive expressions. By combining Handel-C macros (those defined with macro expr) and the select(. . .) operator discussed above, recursive macros can be used to simply express complex hardware. This type of macro is particularly important in Handel-C where the exact form of the macro may depend on the width of a parameter to the macro. As an example, a sign extension of a variable is taken. When assigning a narrow signed variable to a wider variable, the most significant bits of the wide variable should be padded with the sign bit (MSB) of the narrow variable. For example, the 4-bit representation of -2 is 0b1110. When assigned to an 8-bit wide variable, this should become 0b11111110. In contrast, the 4-bit representation of 6 is 0b0110. When assigned to an 8-bit wide variable, this should become 0b00000110. In this example, the following code would suffice:

Detail Description Paragraph (4383):
[4496] This example writes the two expressions a+b and c*d twice to the channel out. This example also illustrates that the statement may be a code block--in this case two instructions executed sequentially. Macro procedures generate the hardware for their statement every time they are referenced. The above example expands to 4 channel output statements. Macro procedures differ from preprocessor macros in that they are not simple text replacements. The statement section of the definition may be a valid Handel-C statement. For example:

Detail Description Paragraph (4393):
[4506] This is a valid preprocessor macro definition. However, the following code is not allowed:

Detail Description Paragraph (6356):
[6468] The C preprocessor is a macro processor that is used automatically by the C compiler to transform the program before actual compilation. It is called a macro processor because it allows one to define macros, which are brief abbreviations for longer constructs.

Detail Description Paragraph (6487):
[6599] If one uses the macro name followed by something other than an open-parenthesis (after ignoring any spaces, tabs and comments that follow), it is not a call to the macro, and the preprocessor does not change what he or she has written. Therefore, it is possible for the same name to be a variable or function in the program as well as a macro, and one can choose in each instance whether to refer to the macro (if an actual argument list follows) or the variable or function (if an

argument list does not follow).

Detail Description Paragraph (6404):
[6485] Macros can be defined or undefined with `-D` and `-U` command options when one compiles the program. One can arrange to compile the same source file into two different programs by choosing a macro name to specify which program one want, writing conditionals to test whether or how this macro is defined and then controlling the state of the macro with compiler command options.

Detail Description Paragraph (6593):
[6673] FIG. 87E illustrates a method 8780 for generating libraries utilizing pre-compiler macros, in accordance with one embodiment of the present invention. In general, in operation 8782, a library is accessed that includes a plurality of functions. A precompiler constant is tested in operation 8784 so that one or more of the functions of the library can be selected based on the testing. Note operation 8786.

Detail Description Paragraph (6596):
[6676] In use, a library with an unknown in it can be passed therein at compile time to execute different functions. When a bit size goes above a certain level, one may have to be able to process it differently. As such, a library is created containing different compile time functions as separate macros. Users can set which macro is executed based on the state of the system by testing a precompiler constant. Further, the pre-compiler macro may be used to select which version is utilized.

Detail Description Paragraph (6775):
[6855] This is pre-compiler macro, it should be used at global scope. It should be used after the header is included for the application layer library. It performs the necessary definitions and declarations required for the user core.

Detail Description Paragraph (6881):
[6961] This is a pre-compiler macro and should be used at global scope. It declares a read port for auxiliary I/O. This macro may write a function that provides the functionality to read the named I/O port. Access to the function is provided by the CoProcessorAuxRead macro.

Detail Description Paragraph (6886):
[6966] This is a pre-compiler macro and should be used at global scope. It declares a write port for auxiliary I/O. This macro may write a function that provides the functionality to write to the named I/O port. Access to the function is provided by the CoProcessorAuxWrite macro.

Detail Description Paragraph (6891):
[6971] This is a pre-compiler macro and should be used at global scope. It declares a port for auxiliary I/O that is bidirectional. This macro may write functions that provide the functionality to read, write and set the output buffer mode for the named I/O port. Access to the function is provided by the CoProcessorAuxRead, CoProcessorAuxWrite and CoProcessorAuxSetEnable macros.

Detail Description Paragraph (6918):
[6998] This is a pre-compiler macro that can be used anywhere. It is a utility macro that allows access to the width of an auxiliary 1/0 port. This is useful when defining variables that connect to a port.

Detail Description Paragraph (6927):
[7007] This is a pre-compiler macro. It should only be used at global scope. It may construct the necessary connection to the local bus and any other platform specific definitions.

Detail Description Paragraph (6937):
[7017] This is a pre-compiler macro. It should only be used at global scope. This macros may be used to configure the clock source for a user core. It may be possible for a user core to be clocked at a different rate to the physical core. This is only possible if the physical layer library for the target platform provides more than one clock source.

Detail Description Paragraph (6960):
[7040] This macro is a pre-compiler macro. It should only be used in global scope. It constructs a set of functions that form a memory management unit for the named

memory bank. The method of memory management used is single port exclusive therefore the MMU is a simple transaction sequencer.

Detail Description Paragraph (6967):
[7047] This macro is a pre-compiler macro. It should only be used in global scope. It constructs a set of functions that formn a memory management unit for the named memory bank. A multi-port MMU is constructed that sequences memory requests and provides simple arbitration for the available ports. Semaphores are created and the access functions are constructed as an array of functions.

WEST

Generate Collection | Print

L3: Entry 8 of 9 File: PGPB Feb 6, 2003

DOCUMENT-IDENTIFIER: US 20030028864 A1
TITLE: System, method and article of manufacture for successive compilations using incomplete parameters

Brief Description of Drawings Paragraph (101):
[0106] FIG. 87E illustrates a method for generating libraries utilizing pre-compiler macros, in accordance with one embodiment of the present invention;

Detail Description Paragraph (2082):
[2171] The system described in this example consists of a ROM containing the program to execute, a RAM containing some scratch variables and a processor that understands 10 opcodes. Each instruction is made up of a 4 bit opcode and a 4 bit operand. The _asm_ preprocessor macro is the assembler for this language and is used to fill in the entries in the program ROM declaration.

Detail Description Paragraph (3837):
[3924] A serious limitation with preprocessor macros (those defined with #define) is their inability to generate recursive expressions. By combining Handel-C macros (those defined with macro expr) and the select(. . .) operator discussed above, recursive macros can be used to simply express complex hardware. This type of macro is particularly important in Handel-C where the exact form of the macro may depend on the width of a parameter to the macro. As an example, a sign extension of a variable is taken. When assigning a narrow signed variable to a wider variable, the most significant bits of the wide variable should be padded with the sign bit (MSB) of the narrow variable. For example, the 4-bit representation of -2 is 0b1110. When assigned to an 8-bit wide variable, this should become 0b11111110. In contrast, the 4-bit representation of 6 is 0b0110. When assigned to an 8-bit wide variable, this should become 0b00000110.

Detail Description Paragraph (3942):
[4029] This example writes the two expressions a+b and c*d twice to the channel out. This example also illustrates that the statement may be a code block--in this case two instructions executed sequentially. Macro procedures generate the hardware for their statement every time they are referenced. The above example expands to 4 channel output statements. Macro procedures differ from preprocessor macros in that they are not simple text replacements. The statement section of the definition may be a valid Handel-C statement. For example:

Detail Description Paragraph (3952):
[4039] This is a valid preprocessor macro definition. However, the following code is not allowed:

Detail Description Paragraph (5911):
[5992] The C preprocessor is a macro processor that is used automatically by the C compiler to transform the program before actual compilation. It is called a macro processor because it allows one to define macros, which are brief abbreviations for longer constructs.

Detail Description Paragraph (6043):
[6124] If one uses the macro name followed by something other than an open-parenthesis (after ignoring any spaces, tabs and comments that follow), it is not a call to the macro, and the preprocessor does not change what he or she has written. Therefore, it is possible for the same name to be a variable or function in the program as well as a macro, and one can choose in each instance whether to refer to the macro (if an actual argument list follows) or the variable or function (if an

argument list does not follow).

Detail Description Paragraph (6846):
[6958] Macros can be defined or undefined with `-D` and `-U` command options when one compiles the program. One can arrange to compile the same source file into two different programs by choosing a macro name to specify which program one want, writing conditionals to test whether or how this macro is defined and then controlling the state of the macro with compiler command options.

Detail Description Paragraph (7035):
[7146] FIG. 87E illustrates a method 8780 for generating libraries utilizing pre-compiler macros, in accordance with one embodiment of the present invention. In general, in operation 8782, a library is accessed that includes a plurality of functions. A precompiler constant is tested in operation 8784 so that one or more of the functions of the library can be selected based on the testing. Note operation 8786.

Detail Description Paragraph (7046):
[7157] In use, a library with an unknown in it can be passed therein at compile time to execute different functions. When a bit size goes above a certain level, one may have to be able to process it differently. As such, a library is created containing different compile time functions as separate macros. Users can set which macro is executed based on the state of the system by testing a precompiler constant. Further, the pre-compiler macro may be used to select which version is utilized.

Detail Description Paragraph (7258):
[7369] This is pre-compiler macro, it should be used at global scope. It should be used after the header is included for the application layer library. It performs the necessary definitions and declarations required for the user core.

Detail Description Paragraph (7386):
[7497] This is a pre-compiler macro and should be used at global scope. It declares a read port for auxiliary I/O. This macro may write a function that provides the functionality to read the named I/O port. Access to the function is provided by the CoProcessorAuxRead macro.

Detail Description Paragraph (7392):
[7503] This is a pre-compiler macro and should be used at global scope. It declares a write port for auxiliary I/O. This macro may write a function that provides the functionality to write to the named I/O port. Access to the function is provided by the CoProcessorAuxWrite macro.

Detail Description Paragraph (7398):
[7509] This is a pre-compiler macro and should be used at global scope. It declares a port for auxiliary I/O that is bi-directional. This macro may write functions that provide the functionality to read, write and set the output buffer mode for the named I/O port. Access to the function is provided by the CoProcessorAuxRead, CoProcessorAuxWrite and CoProcessorAuxSetEnable macros.

Detail Description Paragraph (7428):
[7539] This is a pre-compiler macro that can be used anywhere. It is a utility macro that allows access to the width of an auxiliary I/O port. This is useful when defining variables that connect to a port.

Detail Description Paragraph (7437):
[7548] This is a pre-compiler macro. It should only be used at global scope. It may construct the necessary connection to the local bus and any other platform specific definitions.

Detail Description Paragraph (7447):
[7558] This is a pre-compiler macro. It should only be used at global scope. This macros may be used to configure the clock source for a user core. It may be possible for a user core to be clocked at a different rate to the physical core. This is only possible if the physical layer library for the target platform provides more than one clock source.

Detail Description Paragraph (7478):
[7589] This macro is a pre-compiler macro. It should only be used in global scope. It constructs a set of functions that form a memory management unit for the named

memory bank. The method of memory management used is single port exclusive therefore the MMU is a simple transaction sequencer.

Detail Description Paragraph (7486):
[7597] This macro is a pre-compiler macro. It should only be used in global scope. It constructs a set of functions that form a memory management unit for the named memory bank. A multi-port MMU is constructed that sequences memory requests and provides simple arbitration for the available ports. Semaphores are created and the access functions are constructed as an array of functions.

WEST

End of Result Set

☐ Generate Collection Print

L4: Entry 1 of 1 File: PGPB Nov 28, 2002

DOCUMENT-IDENTIFIER: US 20020178290 A1
TITLE: Method and system for converting user interface source code of a legacy application to web pages

Detail Description Paragraph (24):
[0046] Without a doubt, source-to-source conversion refines the generated output. Users not happy with the resulting output have two choices to improve it: (a) edit the generated output source directly using any of the industry JavaServer Page editors such as IBM's WebSphere Studio; or (b) extend the conversion algorithm by writing Java code and offering alternative implementations for the specific construct, e.g., record or field or keyword being converted. The second alternative is easily enabled by offering published extension points in the algorithm in the form of non-final Java classes which can be extended and/or Java interfaces (alternative Java class implementations can be supplied). These extensions are easily achievable because Java is an open language that is easily accessible on any platform and because it is an object oriented language supporting refinement through inheritance and interfaces. By contrast, refining the converted output from the legacy application data stream at runtime is very tedious and difficult and is limited by the amount of information available in the data stream. To do this requires a tool to tell the runtime intercept how to identify the construct to be converted differently and then how to render or convert it to the alternative result. This usually results in runtime macros that are executed by the conversion code at runtime. On many thousands of screens this can be very time consuming and error prone. Further, it can further degrade the runtime performance as there is additional runtime overhead to find the user macros and run those macros, possibly for every record format in the legacy application datastream.

WEST

Help | Logout | Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences | Cases

Search Results -

Terms	Documents
L3 and macro same language	0

Database: US Patents Full-Text Database; US Pre-Grant Publication Full-Text Database; JPO Abstracts Database; EPO Abstracts Database; Derwent World Patents Index; IBM Technical Disclosure Bulletins

Search: Refine Search

Recall Text | Clear

Search History

DATE: Thursday, July 31, 2003 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L5	L3 and macro same language	0	L5
L4	L3 and macro	1	L4
L3	L2 and (java or c or c++ or fortran or cobol)	34	L3
L2	uncompiled near code	37	L2
L1	uncompiled near programming near code	0	L1

END OF SEARCH HISTORY

WEST

Help | Logout | Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences | Cases

Search Results -

Terms	Documents
L1 and macro	98

Database: US Patents Full-Text Database; US Pre-Grant Publication Full-Text Database; JPO Abstracts Database; EPO Abstracts Database; Derwent World Patents Index; IBM Technical Disclosure Bulletins

Search: Refine Search

Recall Text | Clear

Search History

DATE: Thursday, July 31, 2003 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L3	L1 and macro	98	L3
L2	L1 and macro near command	0	L2
L1	java near programming near language	1854	L1

END OF SEARCH HISTORY